EV369763323

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Glitch-Free Realtime Playback

Inventor(s):

Gareth Alan Howell

Olivier Colle

Rebecca C. Weiss

421 West Riverside, Suite 500
Spokane, WA 99201
P: 509.324-9256
F: 509.323-8979
www.leehayes.com

lee&hayes

ATTORNEY DOCKET NO.: MS1-1927us

GLITCH-FREE REALTIME PLAYBACK

TECHNICAL FIELD

[0001] This invention generally relates to a technology for facilitating playback of multimedia (e.g., video and audio).

BACKGROUND

[0002] Many video editing products exist for capturing and editing multimedia content, such as video. Many of these products are software running on computing systems. Many of these products allow the user to render a final product for storage on tape, hard drive, CD-ROM, and DVD or for distribution over a network.

[0003] With these products, a user typically captures several video clips. When editing, the user builds a timeline of video clips, which the user has selected from the captured clips. At this point, it is a collection of video clips strung together on a timeline.

[0004] Transitions

[0005] Sometimes simple cuts from one clip to the next work well, but sometimes the cut is harsh and unappealing. Therefore, many video editing products offer the user the option to include a transition from one clip to another. Common examples of transitions include dissolves, wipes, and fades. Often using a drag-and-drop technique, a user selects a transition and the two clips to

transitions between. For example, if the user chooses a fade-out/fade-in transition, clip one will fade-out and then clip two will fade-in.

[0006] The user needs not calculate how to perform such transitions. Rather, when the completed "movie" is rendered from the editing application, the video editing product calculates and renders the transitions from one clip to the next.

[0007] However, the user typically wants to see how this transition will appear without rendering the entire movie and encoding/archiving to a separate file—which can be time-consuming. Indeed, it is quite common for the user to want to see the transition nearly immediately after placing it into timeline of clips.

[0008] **Glitchy Realtime Playback**

[0009] If the user immediately watches just the portion of the movie with the new transition therein, the user might experience glitchy realtime playback. This is because the system is attempting to apply the transition in real-time during playback, when in fact the computations involved are too complex to be done in real time.

[0010] In those situations, user will see a glitch, interruption, jumpiness, or jerkiness in the video playback, or the video might just play back too slowly. In other words, the playback will not be smooth. Therefore, the user may believe that some error has occurred in the creation of the movie, but, in reality, it is only that the transitions cannot be computed on the fly in real time during playback.

[0011] It is a "realtime" playback because the intent is for the computer system to display the rendered playback video segment at the same rate or, at least, no slower than it decodes and renders it. Indeed, there is typically a normal decode schedule that specifies the normal rate for decoding a segment of video. Often the rendering and displaying a video segment matches the decode schedule. The realtime playback has a glitch when rate of rendering or displaying a video segment exceeds the rate of decoding that segment.

[0012] With some systems and in some situations, the momentary processing requirements for generating the newly inserted transitions between scenes and the processing requirements for realtime playing back the video may exceed the total available processing capacity.

[0013] In other words, the decoding and rendering of a video segment (including its transition) may require more than all of the available computing resources on a computer system at a given moment. An example of such a situation when one is attempting to play back a transition between two encoded video clips.

[0014] The chances of such a glitch occurring during a realtime playback increase along with the overall bandwidth of the video clips (which includes resolution and number of frames per second) and the complexity of the transform being applied. This is particularly true as high-definition (HD) television and video equipment is becoming increasingly more common.

[0015] The chances of a glitch also increase with the complexity of the transition being performed. It decreases with the overall processing power of the computer system doing the playback.

[0016] **Conventional: Glitch-Free Playback**

[0017] Some conventional video editing products do nothing to avoid the possibility of a glitchy realtime playback. They simply hope that computer processing power will exceed the processing requirements of realtime playback. However, as indicated above, that "hope" is slowing fading because the overall bandwidth of video is increasing (especially with the advent of HD video onto the market).

[0018] Some conventional products attempt to avoid the problem of glitchy playback by introducing a delay that removes the "realtime" aspect of the playback.

[0019] These conventional products pre-render the segment of video having the transitions therein. In other words, the product begins rendering the video segment having the transitions in anticipation of the user wishing to view that segment. So when playing back that segment, it shows the already pre-rendered segment.

[0020] "Pre-rendering" refers to writing a video segment to disk with transitions and effects applied so that this output can be played back without having to apply those transitions and effects in real time. Pre-rendering is a time-consuming operation that interrupts the users workflow.

[0021] To allow for pre-rendering, the user's playback is delayed. Therefore, it no longer realtime playback. Instead, the playback is based, at least in part, on pre-processing.

[0022] While that delay introduced by pre-rendering may be short, the user is prohibited from immediate playback. Indeed, that pre-rendering delay will only increase as the video bandwidth and the complexity of the transition does.

[0023] No conventional video editing product allows for immediate playback of video segments (with transitions applied therein) which is both glitch-free and realtime.

SUMMARY

[0024] Described herein is a technology for facilitating playback of video segments (with transforms applied therein) which can be both glitch-free and realtime.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] The same numbers are used throughout the drawings to reference like elements and features.

[0026] Fig. 1 is a schematic block diagram in accordance with an implementation described herein.

[0027] Fig. 2 is a flow diagram showing an illustrative methodological implementation described herein.

DETAILED DESCRIPTION

[0028] The following description sets forth techniques that facilitate immediate playback of video segments (with transforms applied therein) which can be both glitch-free and realtime. The techniques may be implemented in many ways, including (but not limited to) via program modules, on computer systems, dedicated electronics (such as multimedia appliances) and on computer networks.

[0029] An exemplary implementation of these techniques may be referred to as an "exemplary realtime playback system" and is described below.

[0030] **Exemplary Realtime Playback System**

[0031] Fig. 1 illustrates at least a portion of a video editing system 100. That portion implements the exemplary realtime playback system. It may be implemented as a program module which operates on a computing system. For example, it may be part of video editing software product or it may be implemented as part of a video editing appliance.

[0032] With the video editing system 100, a user builds a timeline of captured multimedia clips 105, such as video clips. Such clips may also be called multimedia signals. The user may insert a transition (e.g., dissolves, wipes, and fades) between clips. Alternatively, the user may apply some effect some segment of the video (regardless of whether two clips border each other). An example of a video effect is "Film Age" (to give it that old–fashioned look).

[0033] These transitions and effects are called "transforms" herein. The various transforms that the video editing system 100 makes available may be stored in a database 115 of such transforms. Other examples of transforms include titling, transitions, picture-in-picture, effects, or any two or three dimensional video transformation.

[0034] To playback a video segment with a just-applied transform ("transformed video segment" herein), the multimedia decoder 110 decodes the video segment. That segment may include first video clip followed by a second one.

[0035] At a designated point in the decoding of the video segment, the transformer 120 calculates the effect of the transform (such as a transition between clips) on the video segment. The renderer 130 renders the decoded (and transformed) video segment and displays it on display 150.

[0036] On a conventional system, when the above actions (of decoding, transforming, and rendering) are performed in realtime, the resulting playback often stutters and skips. This is often called a "glitch."

[0037] For example, in a conventional scenario, a typical 720x480 video clip can take anywhere from 40%-55% of the available computing resources to play back, and the transition may take another 10-15% of the resources.

[0038] So with two clips and the transition, approximately 90%-125% of the available resources are required to perform a realtime playback the video

segment without a glitch in a conventional scenario. This means that it is likely that conventional realtime playback will glitch.

[0039] Examples of computing resources include CPU cycles, memory, and system bus bandwidth. Also, the focus here is only "available" resources. Resources dedicated to other computing tasks are effectively unavailable.

[0040] The exemplary realtime playback system takes advantage of the fact that there are spare computing resources available while playing back the timeline in realtime. For example, there are spare resources available while playing back the first video clip since it only takes 40%-55% of the resources to play back that clip.

[0041] While the first video clip is playing back, the decoder 110 of the video editing system 100 utilizes those otherwise unused computing resources to decode ahead in the timeline and get a few seconds ahead of playback. This "decoding ahead" during the realtime playback effectively smoothes over the temporary peaks in usage of computing resources. This allows the timeline to playback glitch-free and in realtime.

[0042] Since the decoder 100 is decoding ahead, the transformer 120 is able to effectively transform ahead as well. In some instances, there may not be any transform being applied. If so, the decoded-ahead portion of the video segment by-passes or passes-through the transformer 120.

[0043] The video segment that is decoded (and possibly transformed) ahead of when it is needed is stored in one or more buffers 140. It may be called a

"decode-ahead" buffer. The renderer 130 receives the decoded-ahead portion of the video segment when such portion is scheduled for rendering and display. The renderer 130 is typically implemented on a hardware video card and includes video memory (VRAM) 132. In one or more implementations described herein, the decode-ahead buffer 140 may reside in VRAM 132 or in the system memory of a computing system.

[0044] VRAM is used to store image data for a computer display. Video RAM is really a buffer between the computer processor and the display and is often called the frame buffer. When images are to be sent to the display, the image data is written to VRAM. From VRAM, the image data is converted by a RAM digital-to-analog converter (RAMDAC) into analog signals that are sent to the display presentation mechanism, such as a cathode ray tube (CRT).

[0045] Typically, VRAM is located on dedicated video processing hardware (such as video card) in the computer. Most forms of VRAM are dual-ported, which means that while a processor is writing a new image to VRAM, the display is reading from VRAM to refresh its current display content. The dual-port design is one of the main differences between system memory and video RAM.

[0046] With exemplary realtime playback system, the user is able to immediately (without any delay) playback a transformed video segment in realtime and does so glitch-free. Unlike the conventional approaches, there does not need to be a delay instituted to pre-render the transformed video segment. The exemplary realtime playback system effectively accumulates a head start while

playing back the video segment. It takes advantage of times when less than all of the available resources are utilized to get ahead.

[0047] Alternatively, this head start is obtained up-front, wherein the actual display of the video segment is delayed a very brief amount of time (on the order of a second or two) while the exemplary realtime playback system buffers decoded-ahead portion of video. This has the cost of a slightly increased startup time.

[0048] **Exemplary Embodiments**

[0049] This section describes several examples of embodiments of the exemplary realtime playback system. Of course, these are just examples and other embodiments are within the scope of the appended claims.

[0050] In one embodiment, the decode-ahead buffer 140 is located in the primary system memory of a computing system. It is often simply called RAM (Random Access Memory) and it is typically accessed via a system bus.

[0051] In another embodiment, the decode-ahead buffer 140 is located within the VRAM 132 of the renderer 130. This allows for particularly quick read/write times and quick data transfer for the renderer 130.

[0052] Also, this allows for the GPU (graphic card CPU) to be used for all video transforms. Let's assume that one stores in the system memory of the computing system instead in a VRAM, but that one uses the GPU for the transforms. In this case, one would have to copy the input frames into the VRAM, perform the transition using the GPU, and then bring back the resulting frame into

the system memory. This last operation is extremely costly in terms of computing resources and time. By storing into VRAM, one avoids this last step, and takestake advantage of the GPU for all video transforms.

[0053] In still another embodiment, the transformer 120 is co-located with the renderer 130 on dedicated video hardware and the decode-ahead buffer 140 is located within the VRAM 132 on the hardware. Some dedicated video hardware offer hardware accelerated transformations. In these instances, the decoder 110 decodes video segment directly into VRAM 132 of the renderer 130. The transformer 120 performs hardware accelerated transformations on the decoded video segments in the VRAM 132 and stores the results in the buffer 140 within the VRAM.

[0054] In yet another embodiment, the decoder 110 and transformer 120 are co-located with the renderer 130 on dedicated video hardware and the decode-ahead buffer 140 is located within the VRAM 132. Some dedicated video hardware offer hardware accelerated transformations and decoding. In these instances, the decoder 110 performs hardware accelerated decoding of the video segment directly into VRAM 132 of the renderer 130. The transformer 120 performs hardware accelerated transformations on the decoded video segments in the VRAM 132 and stores the results in the buffer 140 within the VRAM.

[0055] **Methodological Implementation**

[0056] Fig. 2 shows a methodological implementation of the exemplary realtime playback system). This methodological implementation may be performed in software, hardware, or a combination thereof. For ease of

understanding, the method steps are delineated as separate steps; however, these separately delineated steps should not be construed as necessarily order-dependent in their performance.

[0057] At 210 of Fig. 2, the multimedia decoder 110 obtains one or more multimedia clips.

[0058] At 212, the transformer 120 obtains one or more transforms from a database 115 of such.

[0059] At 214, the decoder 110 decodes the one or more clips. While doing that, the exemplary realtime playback system monitors whether there are spare computing resources available. If so, the decoder 110 increases its rate of decoding to utilize some or all of the otherwise spare resources. Decoding/transforming happens as quickly as computing resources will allow, subject to the constraint of how much decode-ahead buffer space is available. This increased rate of decoding may or may not be the same as the normal decoding rate specified for the video clips being decoded; for simple clips, though, it can be expected to be faster.

[0060] At 216, the transformer 120 receives the one or more decoded clips and applies the one or more transforms to them. For example, it may produce a fade-out from one clip and a fade-in to another one. Since the decoder 110 is supplying decoded clips at a rate greater than normal, the transformer 120 matches (or at least come close to doing so) the increased rate.

[0061] At 218, the decoded-ahead (and possibly transformed-ahead as well) portions of the clips are stored in the decode-ahead buffer 140. The renderer 130 is not yet ready to render these clips.

[0062] At 220, the exemplary realtime playback system supplies the renderer 130 with clips at a normal render/display rate for such clips. That includes supplying decoded-ahead clips from the decode-ahead buffer 140 when they are scheduled to be rendered/displayed.

[0063] At 222, the exemplary realtime playback system renders and displays the supplied clips at a normal rate that the clips are scheduled to be rendered and displayed.

[0064] Note that while the frames of the decode-ahead buffer 140 are being rendered in real time (e.g. 30 fps) , the exemplary realtime playback system is still it filling the buffer. Frame computation happens at the same time as frames are being rendered and, indeed, it will occur:

- when *more* computer resources are available, the frame computation will occur *faster* than frames of the decode-ahead buffer are being rendered; and
- when *less* computer resources are available, the frame computation will occur *slower* than frames of the decode-ahead buffer are being rendered.

[0065] **Conclusion**

[0066] Although the one or more above-described implementations have been described in language specific to structural features and/or methodological

steps, it is to be understood that other implementations may be practiced without the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of one or more implementations.